TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

SPRÁVA SOUBORŮ V PLC

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Lukáš Kin**
Vedoucí práce: Ing. Leoš Beran, Ph.D.

Liberec 2015

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

FILES MANAGEMENT IN PLC

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **Lukáš Kin**
Supervisor: Ing. Leoš Beran, Ph.D.

Liberec 2015

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Lukáš Kin**
Osobní číslo:	**M12000144**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Správa souborů v PLC**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Vytvořte programové vybavení pomocí jazyka ST pro správu souborů v PLC.
2. Jasně definujte příkazy, parametry a stavy programu.
3. Program musí umožnit: správu disků a jejich přepínání; kopírování, mazání, přesouvání a přejmenovávání souborů; vytváření, mazání a přejmenovávání adresářů.
4. Uživatel musí mít možnost procházení jednotlivých disků, adresářů a podadresářů.
5. Výsledky práce musí být pečlivě zdokumentovány včetně nápovědy pro případného uživatele.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **AUTOMATION, B&R. Controls - training text. Austria: [s.n.], 2008. 205 s.**

[2] **BERNECKER + RAINER INDUSTRIE-ELEKTRONIK GES.M. B. H. B&R Help: FileIO. 2013. vyd. 2013.**

[3] **ŠMEJKAL, Ladislav a Marie MARTINÁSKOVÁ. PLC a automatizace. 1. díl. Praha: BEN, 1999. ISBN 80-86056-58-9.**

[4] **JOHN, Kharl-Heinz; TIEGELKAMP, Michael. IEC 61131-3 Programming Industrial Automation Systems : Concepts and Programming Languages, Requirements for Programming Systems, Decision - Making Aids. 2nd Edition. NewYork : Springer, 2010. 390 s. ISBN 978-3-642-12015-2.**

Vedoucí bakalářské práce: **Ing. Leoš Beran, Ph.D.**
Ústav mechatroniky a technické informatiky

Konzultant bakalářské práce: **Ing. Martin Diblík, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2014**

Termín odevzdání bakalářské práce: **15. května 2015**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2014

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Mé poděkování patří Ing. Leoši Beranovi, Ph.D. za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování bakalářské práce věnoval.

Anotace

Tato práce se zabývá vytvořením programu pro správu souboru a adresářů v PLC s použitím knihovny FileIO v softwaru Automation studio od firmy B&R.

Program obsluhuje základní procesy spojené s vytvářením, přejmenování, kopírování, přesouváním a mazáním souborů. Zajišťuje vytváření adresářů, jejich přejmenování a mazání. Provádí práce spojené se správou disků.

Vše s využitím funkčních bloků knihovny FileIO (FileCreate, FileCopy, FileRename, FileDelete, FileClose, Devlink, DevUnlink, DirInfo, DirRead, DirCreate, DirRename, DirDelete, DirDeleteEx)

Klíčová slova

Knihovna FileIO, práce se soubory, práce s adresáři, práce s disky, PLC, Automation Studio, Strukturovaný text

Abstract

This work deals with creating a program for managing files and directories in the PLC using the library FileIO software Automation Studio from B & R.

The program serves the fundamental processes associated with creating, renaming, copying, moving and deleting files. Provides directory creation, renaming and deleting. Performs work associated with managing disk.

Using functional blocks at the library FileIO (FileCreate, FileCopy, FileRename, Filedelete, FileClose, Devlink, DevUnlink, DirInfo, DirRead, DirCreate, DirRename, DirDelete, DirDeleteEx)

Key words

FileIO library, files, work with directories, working with disks, PLC, Automation Studio, Structured Text

Seznam obrázků

Obrázek 1 - Automation Studio 12
Obrázek 2 - Action type 14
Obrázek 3 - Data Type 14
Obrázek 4 - Device Path 14
Obrázek 5 - FunkcionBlock Type 15
Obrázek 6 - ManageFile Type 15
Obrázek 7 – States 16
Obrázek 8 - ManageFile.var 16
Obrázek 9 - Hlavní diagram 17
Obrázek 10 - Obsah disků diagram 19
Obrázek 11 - SetupDevice diagram 20
Obrázek 12 - Správa adresářů - diagram 21
Obrázek 13 - Správa souborů diagram 23

Seznam tabulek

Tabulka 1 - Hlavní diagram 18
Tabulka 2 - Obsah disků 19
Tabulka 3 - SetupDevice 20
Tabulka 4 - Adresářová správa 22
Tabulka 5 - Správa souborů 23

Seznam příloh

Příloha č. 1 - Obsah CD

Seznam zkratek

B&R	Bernecker a Rainer
PLC	Programmable Logic Controller
TUL	Technická univerzita v Liberci
AS	Automation Studio
CNC	Computer Numeric Control
ST	Structured Text
FB	Funkční blok

Obsah

Seznam obrázků 3
Seznam tabulek 3
Seznam příloh 3
Seznam zkratek 3
1. Úvod 10
2. Použité technologie 11
2.1 Automation Studio 11
2.2 Structured Text 12
2.3 ARsim 12
3. Struktura programu 13
3.1 Global.type 13
3.2 Global.var 13
3.3 Libraries 13
3.4 ManageFile 13
4. Stavové diagramy 17
4.1 Hlavní diagram 17
4.2 Obsah disků - diagram 19
4.3 SetupDevice – diagram 20
4.4 Správa adresářů – diagram 21
4.5 Správa souborů – diagram 23
5. Obecné stavy 24
5.1 Inicializace 24
5.2 Obsah disků 24
5.2.1 Dir_Info 24
5.2.2 Read_Dir 25
5.2.3 Read_File 25
5.3 Action 26
5.4 CheckParameter 27
5.5 Return_Path 27
5.6 Dev_Unlink 27
5.7 Dev_Link 27

5.8 Správa disků 28
5.9 ERROR 28
6. Správa adresářů 28
6.1 Open_Dir 28
6.2 Create_Dir 29
6.3 Rename_Dir 29
6.4 Delete_Dir 30
6.5 Delete_Dir_All 30
7. Správa souborů 31
7.1 Create_File 31
7.2 cClose_File 31
7.3 Copy_File 32
7.4 Rename_File 32
7.5 Delete_File 33
8. Závěr 34
Seznam literatury a dalších pramenů 35

1. Úvod

V dnešní době, je automatizace využívána téměř v jakémkoli odvětví průmyslu a její použití je rozmanité. Programy se už ani zdaleka neorientují pouze na ovládání motorů, strojů a jiných elektrických zařízení, které se využívají ve výrobě.

V mnoha odvětvích se pracuje s velkými objemy dat, ať už jsou to měření, záznamy výroby a podobně. Je tedy třeba k tomu vytvořit i odpovídající programové vybavení, které tyto požadavky dokážou zaopatřit.

V této práci byl navržen program pro správu souborů a adresářů v prostředí od společnosti B&R. Společnost nabízí své řešení ve formě MAPP Component, z nichž jedna část se zabývá právě touto problematikou. Bohužel je to však finální produkt, který nelze nijak upravovat či optimalizovat dle našich požadavky.[1]

Právě proto se práce zaměřila na vytvoření programu, který bude variabilní a bude možnost si ho upravit dle vlastního uvážení a potřeby.

V práci se seznámíme s prostředním Automation Studio, ve kterém se program vyvíjí a s jeho součástmi.

Náš program využívá knihovní funkce FileIO, které se starají o procesy spojené se správou souborů a adresářů. Zajišťují jednoduché ovládání a přepínání mezi úložnými zařízeními. Celý program je postaven na jednom stavovém automatu. Program je napsán pomocí programovacího jazyka ST. [2]

2. Použité technologie

Stručný popis prostředků, které byly využity při vytváření bakalářské práce.

2.1 Automation Studio

Je vývojové prostředí společnosti B&R, které je určeno k vývoji aplikací, pomocí kterých jsou ovládány zařízení B&R.

Program je rozložen do tří hlavních celků *Motion, Control a Visualization.*

Prostředí AS je rozloženo do několika částí. Horní lišta, je klasickým ovládacím panelem jaký můžeme najít ve spoustě podobných programech, jsou tam například tlačítka pro vytváření projektů, otevírání projektů, spouštění programu a další různá nastavení. V levé části se nachází blok s možností volby mezi tím, zda je zobrazen *Logical View*, ve kterém je možno spravovat struktura programu (složky, programy). *Configuration View* slouží pro nastavení připojených PLC. Poslední možností je *Physical View* kde je možné připojování a správa periferií, ať už reálných nebo virtuálních. [1]

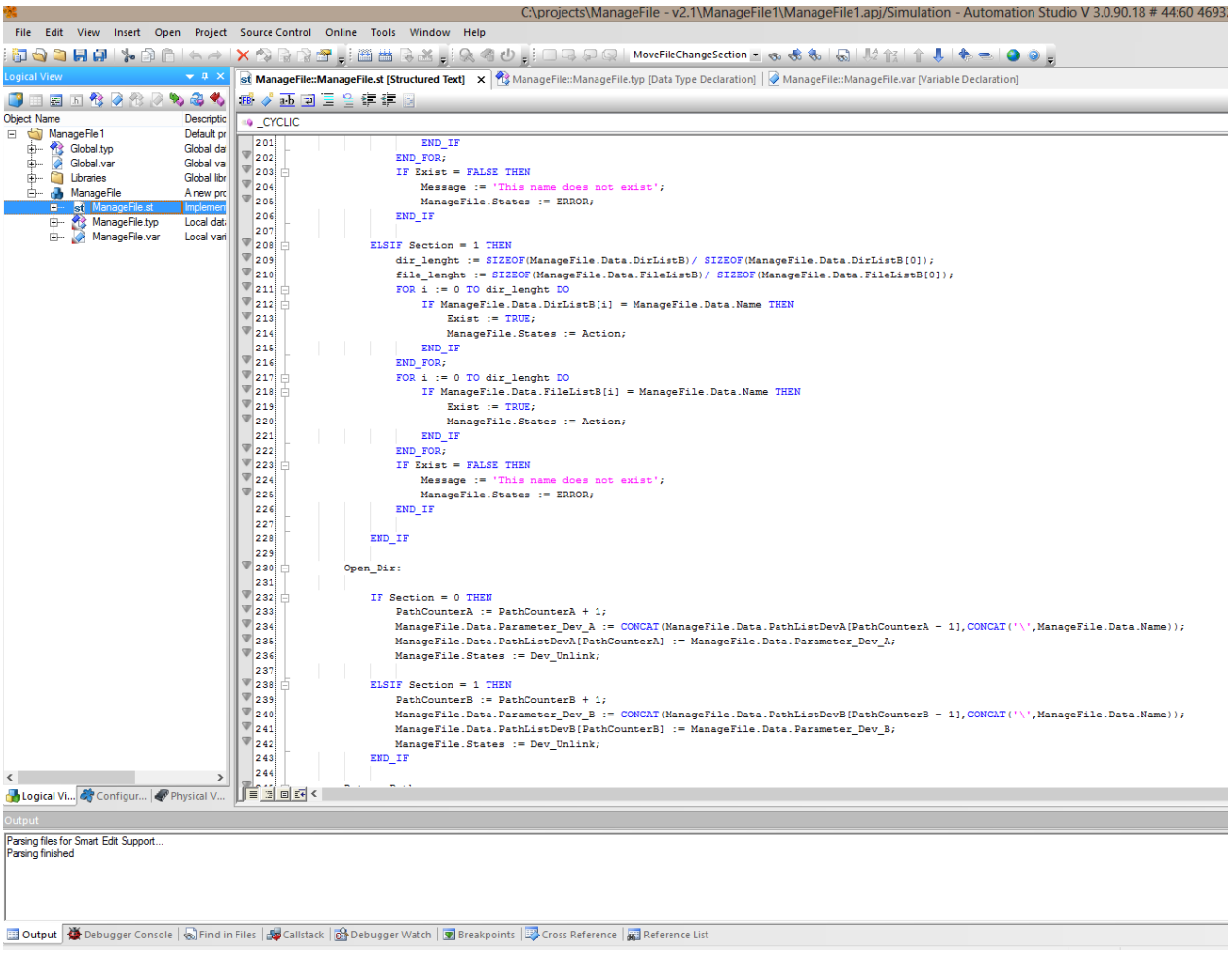
Ve spodní části najdeme sekci, ve které je možnost sledovat probíhající kompilace, chybové hlášení a další informace, které jsou sdělovány za běhu programu.

Poslední částí je pracovní plocha, ve které se otevírají jednotlivé záložky, které reprezentují části našeho programu. V této části se provádí především programování aplikace, ale zobrazují se zde například i výpisy a ladící procesy.

V Automation Studiu je implementována podpora pro mnoho vývojových jazyků až už základních nebo jazyků vyšších úrovní. Vybrat si můžeme například z objektově orientovaných, jako jsou například C++ nebo Structured Text. Další možnosti jsou pak ANSI C, Ladder Diagram. [1]

Ke kontrole funkčnosti programu používáme *Monitor*, který nám umožňuje sledovat stav proměnných, průběh funkcí a jednotlivé části stavových automatů. Při monitoringu je možnost využít vestavěnou součásti pro ladění.

O grafické zobrazení se stará součást *Visualization*, ve které je možnost vytváření GUI pro dotykové panely, vytváření webových stránek pro vzdálenou obsluhu a kontrolu.[1]



Obrázek 1 - Automation Studio

2.2 Structured Text

Jazyk strukturovaného textu (ST) je velmi výkonný objektově orientovaný textový jazyk. Kořeny má ve známých jazycích typu Pascal, Ada, C.

Je velmi účinným nástrojem pro zápis náročných algoritmů a pro vytváření komplexních funkčních bloků. Vyhodnocení výrazu spočívá v aplikování operátorů na operandy s respektováním priorit operátorů. [1]

2.3 ARsim

Je součást AS, která slouží pro simulaci virtuálního PLC na localhostu. Bez potřeby reálného hardware tedy máme možnosti ladění a testování naší aplikace.

Nabízí nám několik možností ovládání jako je restart, vypnutí, diagnostika a zobrazení vizualizace. [1]

3. Struktura programu

Program je třeba si rozdělit do několika částí:

- *Global.type*
- *Global.var*
- *Libraries*
- *ManageFile*

3.1 Global.type

Globální typ umožní vytvořit námi navržené struktury, se kterými můžeme pracovat napříč všemi programy, které jsou obsaženy v projektu.

3.2 Global.var

Globální proměnné jsou viditelné ve všech programech v projektu a dokážeme pomocí nich zajistit komunikaci mezi nimi.

3.3 Libraries

Knihovny obsahují již vytvořené funkční bloky a funkce, které jsou potřebné pro vytváření programů.

Kromě základních knihoven, které jsou připojeny automaticky při vytváření projektu je importována knihovna pro práci s textovými řetězci a kniha s funkčními bloky pro správu disků, souborů a adresářů.

Importované knihovny:

- Runtime – obsahuje funkce pro základní úkony programu
- Operator – obsahuje funkce operátorů (ABS,EXP,AND,OR …)
- AsIecCon – obsahuje funkce pro konverze datových typů
- astime – funkce pro řízení času a data
- asstring – pro práci s textovými řetězci
- standard – obsahuje standardní funkční bloky a funkce pro IEC 1131-3

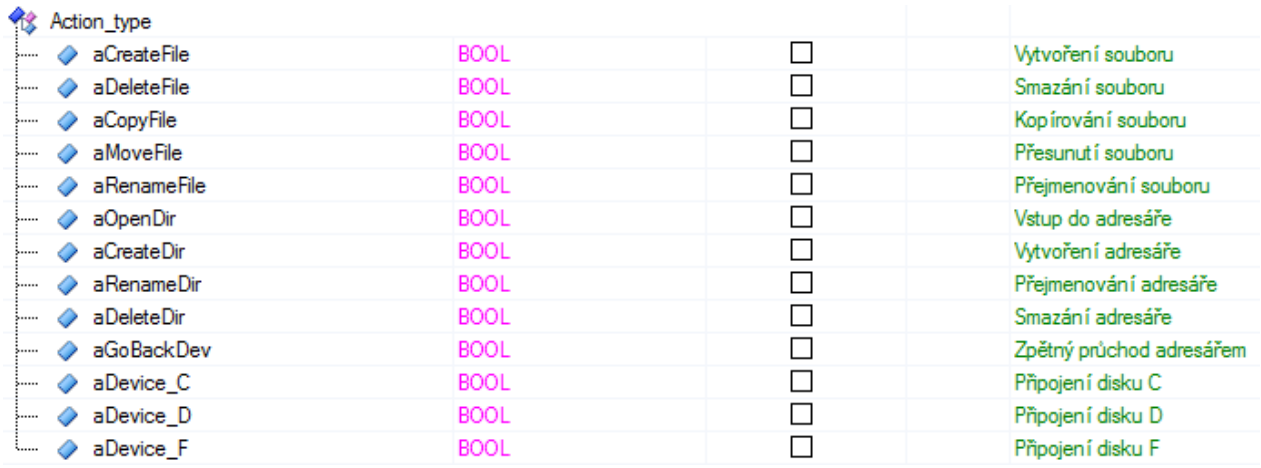
3.4 ManageFile

ManageFile.st obsahuje části *INIT* a *CYCLIC*, ve kterých je napsán aplikační kód.

ManageFile.typ zde je uložena datová struktura pomocí které se přistupuje k jednotlivým definovaným proměnným.

Datová struktura *ManageFile.type* je tvořena ze šesti částí:

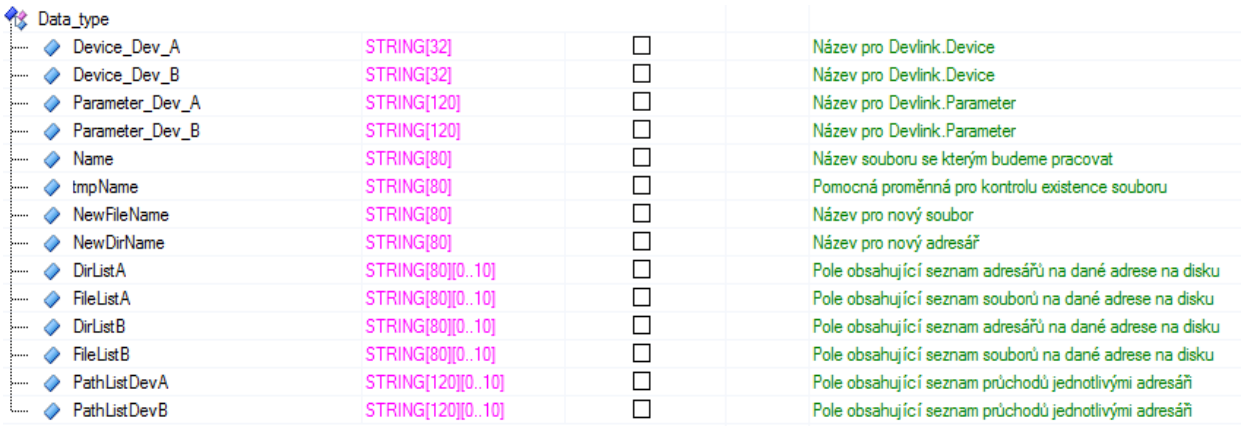
Action_Type zde jsou jednotlivé příkazy, pomocí kterých se vykonávají jednotlivé akce se soubory a adresáři.



Action_type			
aCreateFile	BOOL	☐	Vytvoření souboru
aDeleteFile	BOOL	☐	Smazání souboru
aCopyFile	BOOL	☐	Kopírování souboru
aMoveFile	BOOL	☐	Přesunutí souboru
aRenameFile	BOOL	☐	Přejmenování souboru
aOpenDir	BOOL	☐	Vstup do adresáře
aCreateDir	BOOL	☐	Vytvoření adresáře
aRenameDir	BOOL	☐	Přejmenování adresáře
aDeleteDir	BOOL	☐	Smazání adresáře
aGoBackDev	BOOL	☐	Zpětný průchod adresářem
aDevice_C	BOOL	☐	Připojení disku C
aDevice_D	BOOL	☐	Připojení disku D
aDevice_F	BOOL	☐	Připojení disku F

Obrázek 2 - Action type

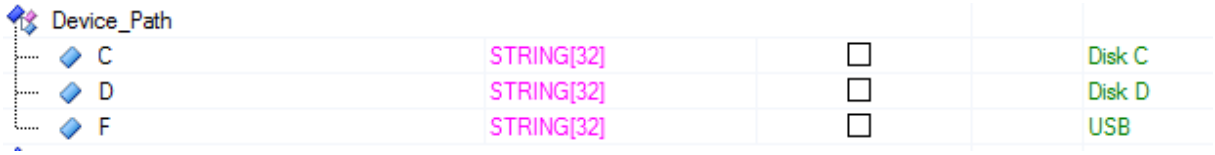
Data_Type tato struktura uchovává ve svých proměnných informace o cestě na disk, název souboru / adresáře a pole s daty.



Data_type			
Device_Dev_A	STRING[32]	☐	Název pro Devlink.Device
Device_Dev_B	STRING[32]	☐	Název pro Devlink.Device
Parameter_Dev_A	STRING[120]	☐	Název pro Devlink.Parameter
Parameter_Dev_B	STRING[120]	☐	Název pro Devlink.Parameter
Name	STRING[80]	☐	Název souboru se kterým budeme pracovat
tmpName	STRING[80]	☐	Pomocná proměnná pro kontrolu existence souboru
NewFileName	STRING[80]	☐	Název pro nový soubor
NewDirName	STRING[80]	☐	Název pro nový adresář
DirListA	STRING[80][0..10]	☐	Pole obsahující seznam adresářů na dané adrese na disku
FileListA	STRING[80][0..10]	☐	Pole obsahující seznam souborů na dané adrese na disku
DirListB	STRING[80][0..10]	☐	Pole obsahující seznam adresářů na dané adrese na disku
FileListB	STRING[80][0..10]	☐	Pole obsahující seznam souborů na dané adrese na disku
PathListDevA	STRING[120][0..10]	☐	Pole obsahující seznam průchodů jednotlivými adresáři
PathListDevB	STRING[120][0..10]	☐	Pole obsahující seznam průchodů jednotlivými adresáři

Obrázek 3 - Data Type

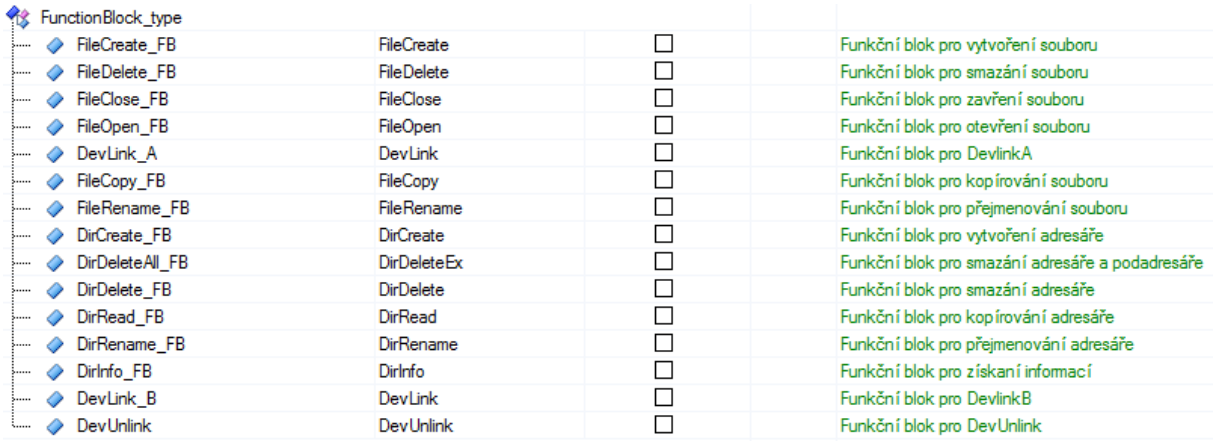
Device_Path proměnné uchovávají cesty pro alokaci disků.



Device_Path			
C	STRING[32]	☐	Disk C
D	STRING[32]	☐	Disk D
F	STRING[32]	☐	USB

Obrázek 4 - Device Path

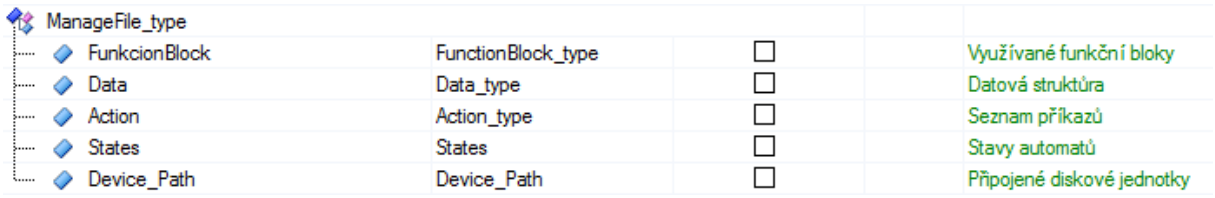
FunkcionBlock_Type seznam funkčních bloků, které zajišťují funkčnost programu.



FunctionBlock_type				
FileCreate_FB	FileCreate	☐		Funkční blok pro vytvoření souboru
FileDelete_FB	FileDelete	☐		Funkční blok pro smazání souboru
FileClose_FB	FileClose	☐		Funkční blok pro zavření souboru
FileOpen_FB	FileOpen	☐		Funkční blok pro otevření souboru
DevLink_A	DevLink	☐		Funkční blok pro DevlinkA
FileCopy_FB	FileCopy	☐		Funkční blok pro kopírování souboru
FileRename_FB	FileRename	☐		Funkční blok pro přejmenování souboru
DirCreate_FB	DirCreate	☐		Funkční blok pro vytvoření adresáře
DirDeleteAll_FB	DirDeleteEx	☐		Funkční blok pro smazání adresáře a podadresáře
DirDelete_FB	DirDelete	☐		Funkční blok pro smazání adresáře
DirRead_FB	DirRead	☐		Funkční blok pro kopírování adresáře
DirRename_FB	DirRename	☐		Funkční blok pro přejmenování adresáře
DirInfo_FB	DirInfo	☐		Funkční blok pro získaní informací
DevLink_B	DevLink	☐		Funkční blok pro DevlinkB
DevUnlink	DevUnlink	☐		Funkční blok pro DevUnlink

Obrázek 5 - FunkcionBlock Type

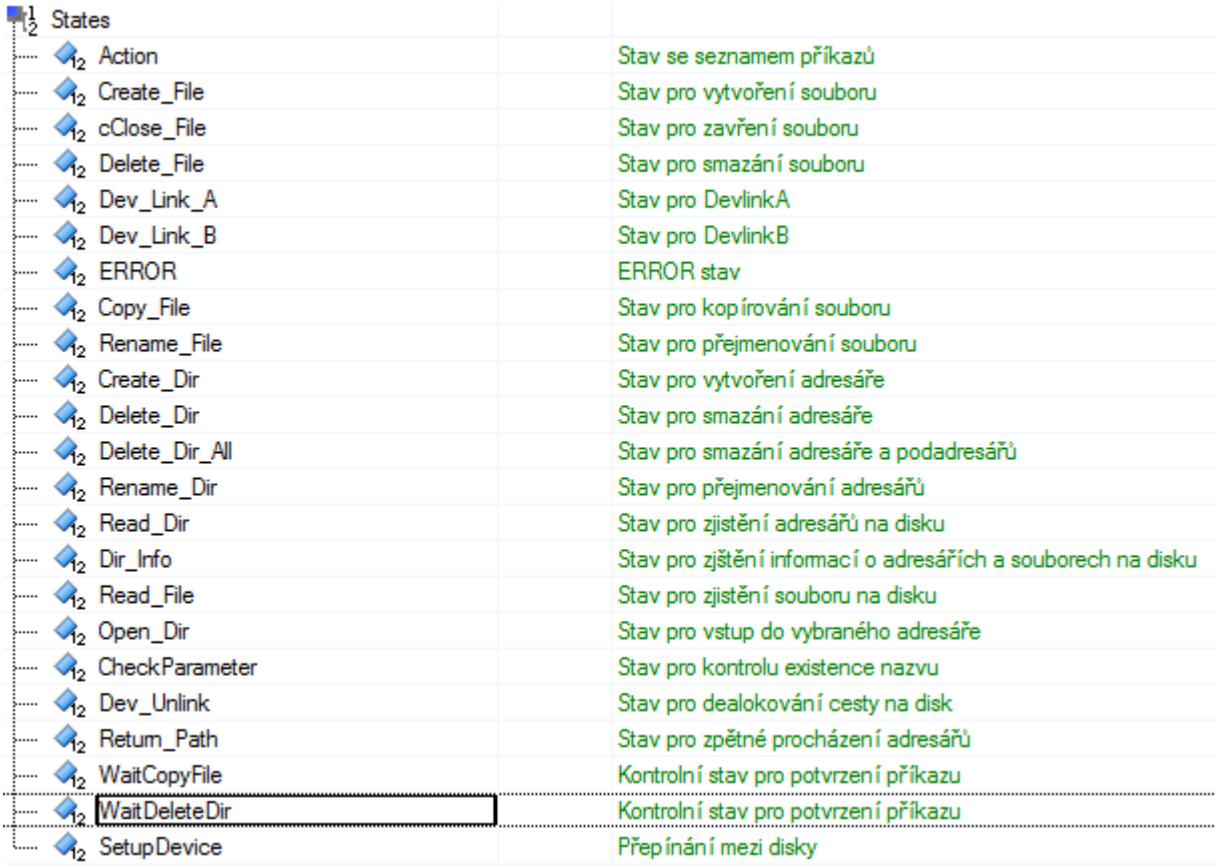
ManageFile_Type rozdělení datové struktury.



ManageFile_type				
FunkcionBlock	FunctionBlock_type	☐		Využívané funkční bloky
Data	Data_type	☐		Datová struktůra
Action	Action_type	☐		Seznam příkazů
States	States	☐		Stavy automatů
Device_Path	Device_Path	☐		Připojené diskové jednotky

Obrázek 6 - ManageFile Type

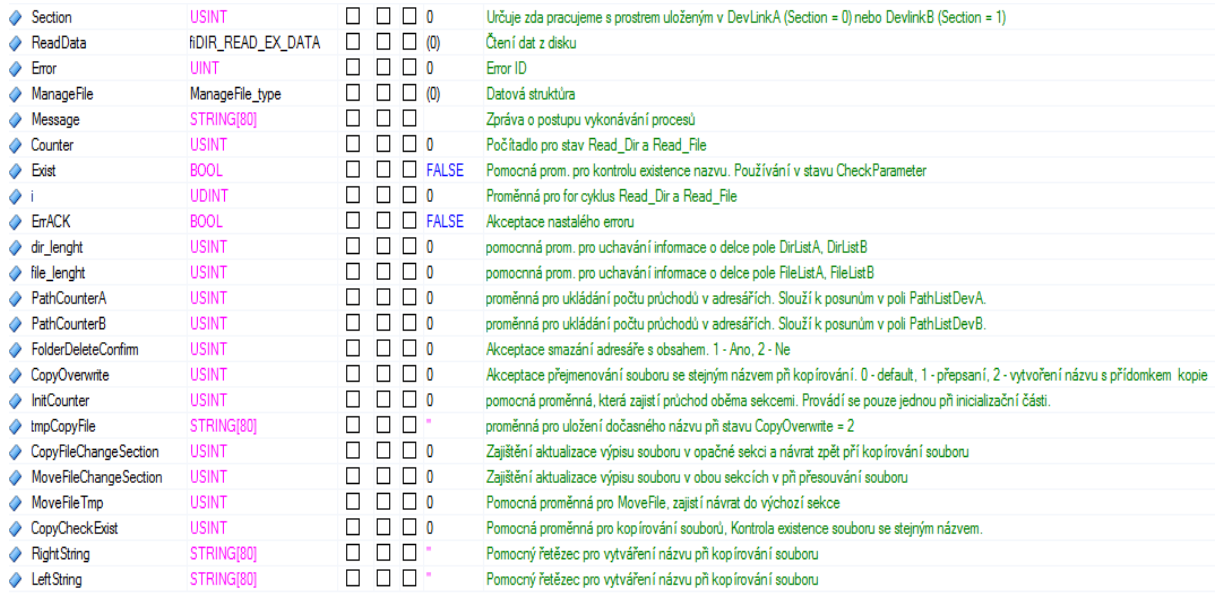
States bloky stavového automatu.



States	
Action	Stav se seznamem příkazů
Create_File	Stav pro vytvoření souboru
cClose_File	Stav pro zavření souboru
Delete_File	Stav pro smazání souboru
Dev_Link_A	Stav pro DevlinkA
Dev_Link_B	Stav pro DevlinkB
ERROR	ERROR stav
Copy_File	Stav pro kopírování souboru
Rename_File	Stav pro přejmenování souboru
Create_Dir	Stav pro vytvoření adresáře
Delete_Dir	Stav pro smazání adresáře
Delete_Dir_All	Stav pro smazání adresáře a podadresářů
Rename_Dir	Stav pro přejmenování adresářů
Read_Dir	Stav pro zjistění adresářů na disku
Dir_Info	Stav pro zjštění informací o adresářích a souborech na disku
Read_File	Stav pro zjistění souboru na disku
Open_Dir	Stav pro vstup do vybraného adresáře
CheckParameter	Stav pro kontrolu existence nazvu
Dev_Unlink	Stav pro dealokování cesty na disk
Return_Path	Stav pro zpětné procházení adresářů
WaitCopyFile	Kontrolní stav pro potvrzení příkazu
WaitDeleteDir	Kontrolní stav pro potvrzení příkazu
SetupDevice	Přepínání mezi disky

Obrázek 7 – States

ManageFile.var zde jsou uchovány proměnné, které nejsou uloženy ve struktuře. Převážně se využívají jako pomocné proměnné pro přechodové stavy nebo pro počítání průchodů.



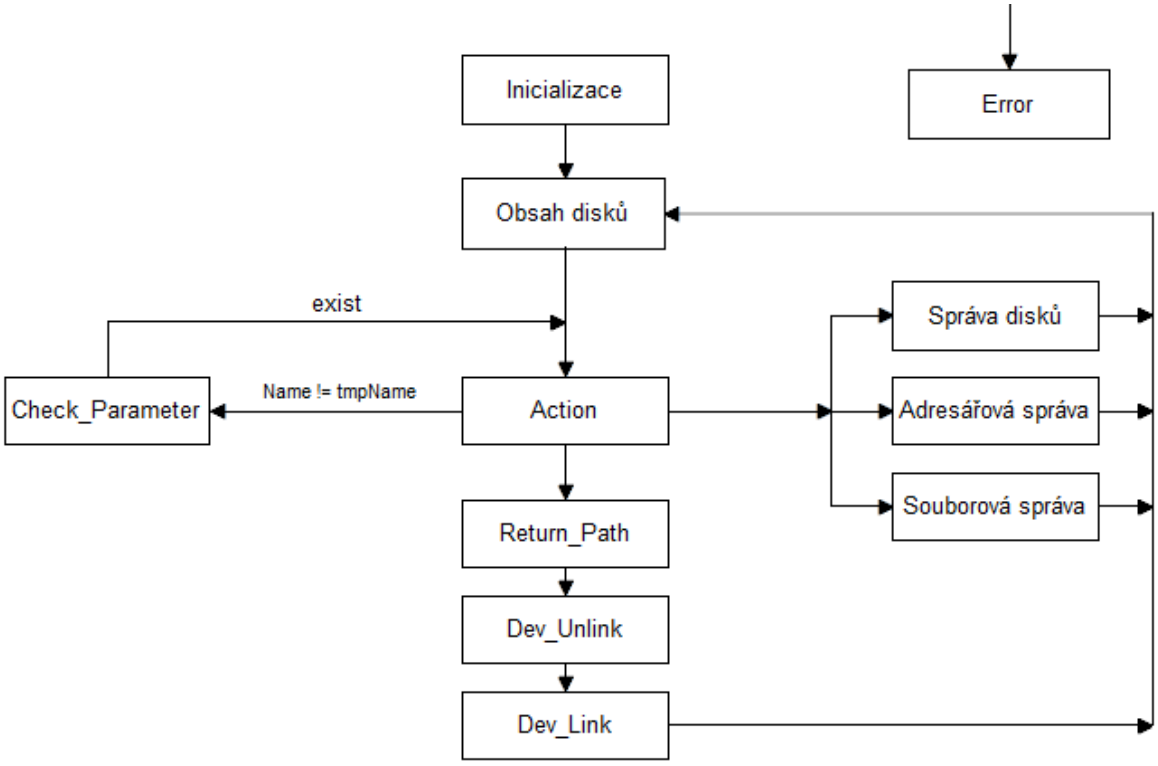
Section	USINT	0	Určuje zda pracujeme s prostrem uloženým v DevLinkA (Section = 0) nebo DevlinkB (Section = 1)
ReadData	fiDIR_READ_EX_DATA	(0)	Čtení dat z disku
Error	UINT	0	Error ID
ManageFile	ManageFile_type	(0)	Datová struktúra
Message	STRING[80]		Zpráva o postupu vykonávání procesů
Counter	USINT	0	Počítadlo pro stav Read_Dir a Read_File
Exist	BOOL	FALSE	Pomocná prom. pro kontrolu existence nazvu. Používání v stavu CheckParameter
i	UDINT	0	Proměnná pro for cyklus Read_Dir a Read_File
ErrACK	BOOL	FALSE	Akceptace nastalého erroru
dir_lenght	USINT	0	pomocnná prom. pro uchavání informace o delce pole DirListA, DirListB
file_lenght	USINT	0	pomocnná prom. pro uchavání informace o delce pole FileListA, FileListB
PathCounterA	USINT	0	proměnná pro ukládání počtu průchodů v adresářích. Slouží k posunúm v poli PathListDevA.
PathCounterB	USINT	0	proměnná pro ukládání počtu průchodů v adresářích. Slouží k posunúm v poli PathListDevB.
FolderDeleteConfirm	USINT	0	Akceptace smazání adresáře s obsahem. 1 - Ano, 2 - Ne
CopyOverwrite	USINT	0	Akceptace přejmenování souboru se stejným názvem při kopírování. 0 - default, 1 - přepsaní, 2 - vytvoření názvu s přídomkem kopie
InitCounter	USINT	0	pomocná proměnná, která zajistí průchod oběma sekcemi. Provádí se pouze jednou při inicializační části.
tmpCopyFile	STRING[80]	''	proměnná pro uložení dočasného názvu při stavu CopyOverwrite = 2
CopyFileChangeSection	USINT	0	Zajištění aktualizace výpisu souboru v opačné sekci a návrat zpět při kopírování souboru
MoveFileChangeSection	USINT	0	Zajištění aktualizace výpisu souboru v obou sekcích v při přesouvání souboru
MoveFileTmp	USINT	0	Pomocná proměnná pro MoveFile, zajistí návrat do výchozí sekce
CopyCheckExist	USINT	0	Pomocná proměnná pro kopírování souborů, Kontrola existence souboru se stejným názvem.
RightString	STRING[80]	''	Pomocný řetězec pro vytváření názvu při kopírování souboru
LeftString	STRING[80]	''	Pomocný řetězec pro vytváření názvu při kopírování souboru

Obrázek 8 - ManageFile.var

4. Stavové diagramy

Grafické zobrazení stavového automatu pomocí diagramů.

4.1 Hlavní diagram

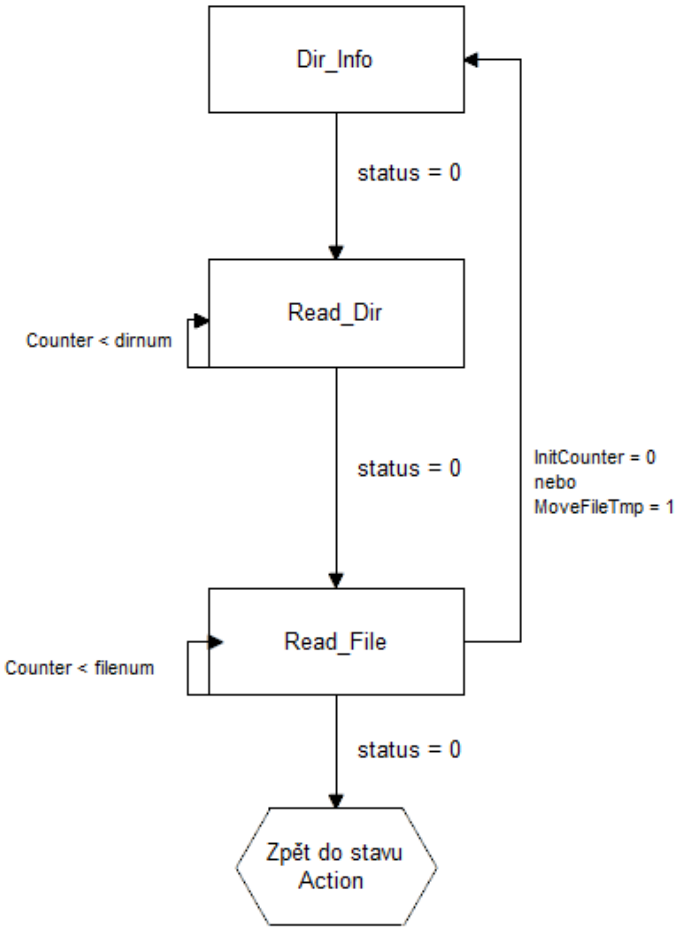


Obrázek 9 - Hlavní diagram

Tabulka 1 - Hlavní diagram

STAV	POUŽITÉ FUNKČNÍ BLOKY	ČINNOST
Inicializace	Devlink, DevUnlink	Počáteční nastavení stavového automatu a načtení adresářů a souborů kořenových adresářů
Obsah Disků		viz. stavový diagram Obsah disků
Action		Ovládací rozhraní
Check_Parameter		Kontrola existence nově zvolené proměnné *Name*
Return_Path		Slouží pro zpětný postup v adresářích. Cesty jsou uloženy v polích průchodů
Dev_Unlink	DevUnlink	Dealokace cesty k místu na disku
Dev_Link	DevLink	Alokace cesty na disk
Správa disků		viz. stavový diagram Správa disků
Adresářová správa		viz. stavový diagram Adresářová správa
Souborová správa		viz. stavový diagram Souborová správa
ERROR		Průchod z jakéhokoliv stavu při error stavu

4.2 Obsah disků – diagram

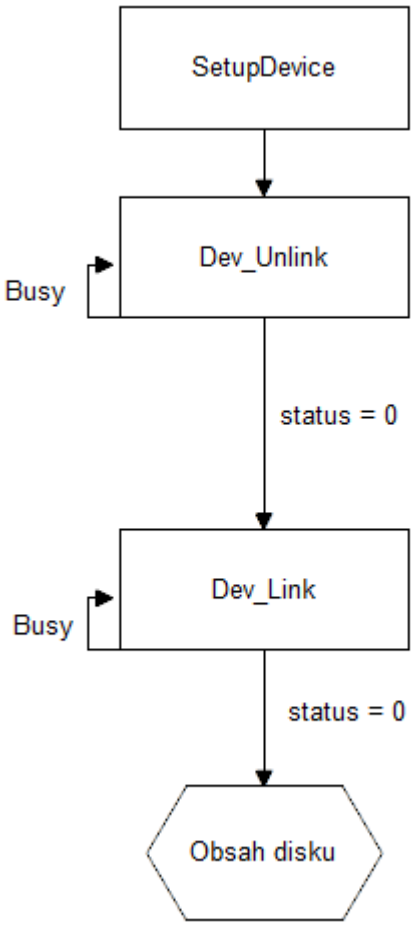


Obrázek 10 - Obsah disků diagram

Tabulka 2 - Obsah disků

STAV	POUŽITÉ FUNKČNÍ BLOKY	ČINNOST
Dir_Info	DirInfo	Zjištění informací o struktuře adresářů a souborů na disku
Read_Dir	ReadDir	Pomocí funkčního bloku s nastaveným parametrem *option = fiDirectory* vložíme do pole strukturu adresářů
Read_File	ReadDir	Pomocí funkčního bloku s nastaveným parametrem *option = fiFile* vložíme do pole strukturu souborů

4.3 SetupDevice – diagram

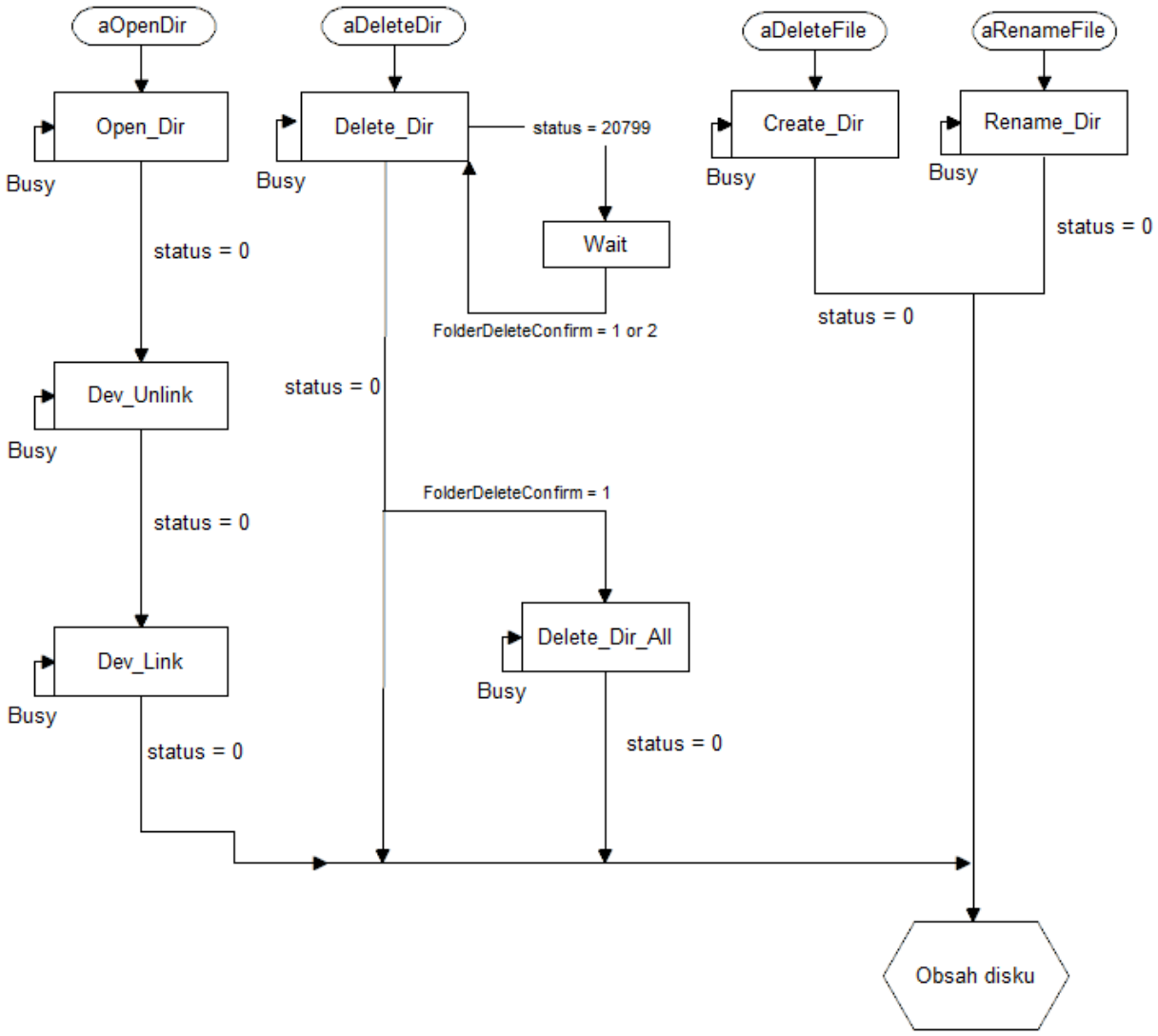


Obrázek 11 - SetupDevice diagram

Tabulka 3 - SetupDevice

STAV	POUŽITÉ FUNKČNÍ BLOKY	ČINNOST
SetupDevice	DirInfo	Nastavení cesty pro kořenový adresář nově připojeného disku
Dev_Unlink	DevUnlink	Dealokace cesty k místu na disku
Dev_Link	DevLink	Alokace cesty na disk

4.4 Správa adresářů – diagram

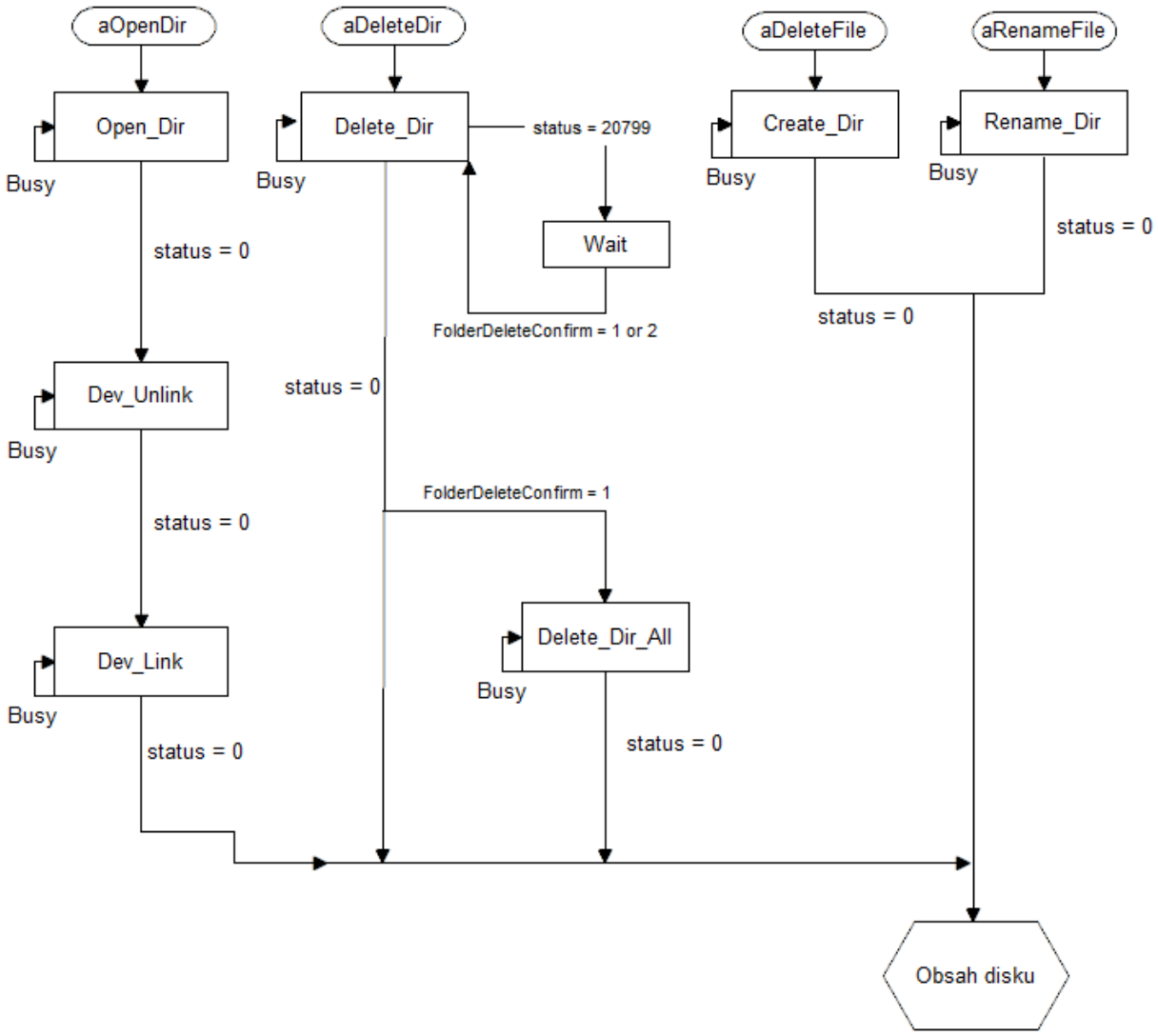


Obrázek 12 - Správa adresářů - diagram

Tabulka 4 - Adresářová správa

STAV	POUŽITÉ FUNKČNÍ BLOKY	ČINNOST
Open_Dir		Vytvoření nové cesty pro parametr funkčního bloku DevLink
Dev_Unlink	DevUnlink	Dealokace cesty k místu na disku
Dev_Link	DevLink	Alokace cesty na disk
Delete_Dir	DirDelete	Smazání vybraného (prázdného) adresáře
Delete_Dir_All	DirDeleteEx	Smazání adresáře i s vnořenými podadresáři a soubory
Create_Dir	DirCreate	Vytvoření nového adresáře se zadaným názvem
Rename_Dir	DirRename	Přejmenování adresáře na zadaný název
Wait		Stav, ve kterém se čeká na interakci uživatele při mazání adresáře, který není prázdný

4.5 Správa souborů – diagram



Obrázek 13 - Správa souborů diagram

Tabulka 5 - Správa souborů

STAV	POUŽITÉ FUNKČNÍ BLOKY	ČINNOST
Create_File	FileCreate	Vytvoření nového souboru se zadaným názvem
cClose_File	FileClose	Zavření nově vytvořeného souboru
Copy_File	FileCopy	Kopírování vybraného souboru. Při příkazu přesunutí souboru dojde ke kopírování souboru a následnému smazání
Delete_File	FileDelete	Smazání vybraného souboru
Rename_File	FileRename	Přejmenování souboru
Wait		Stav, ve kterém se čeká na interakci uživatele při kopírování souboru, když byla zjištěna existence souboru se stejným názvem

5. Obecné stavy

Kapitola popisující inicializační část programu. Jsou zde také uvedeny stavy, které se starají o čtení obsahu z disku, řídí posouvání v adresářové struktuře. Stavy, které ovládají připojování a přepínání disků. Je zde uveden chybový stav ERROR.

5.1 Inicializace

Inicializační část se provede vždy při startu či restartu zařízení.

Probíhá zde počáteční nastavení:

- Kořenový adresářů jednotlivých disků `ManageFile.Device_Path.`
- Pojmenování a přiřazení parametru pro funkční blok DevLink.
- Nastavení kořenového adresáře v poli návratových cest `PathListDevA[0]`, `PathListDevB[0].`
- Nastavení proměnných `PathCounterA := 0; PathCounterB := 0;`
- `Section := 0; InitCounter :=0; CopyFileChangeSection := 2; MoveFileTmp := 0;` na výchozí hodnoty.
- Volání funkčních bloků DevLink pro nastavení počátečních cest k datům.
- Prvotní přesun do stavu `ManageFile.States := Dir_Info;`

5.2 Obsah disků

Klíčový stav, sloužící ke čtení a výpisu adresářové a souborové struktury. Je složen ze tří dílčích stavu *Dir_Info, Read_Dir, Read_File.*

5.2.1 Dir_Info

Do stavu přicházíme z jakéhokoliv stavu, ve kterém dojde ke změně souborů či adresářů nebo při přepínání jednotlivých disků.

Obsahuje funkční blok *DirInfo*, který nám podá základní informace o adresářích a souborech na disku.

Vstupní parametry:

- *Enable* – povolení funkčního bloku
- *pDevice* – ukazatel na název zařízení, které bylo připojeno pomocí FB *DevLink*
- *pPath* – ukazatel na cestu k adresáři

Výstupní parametry:

- *status* – chybové kódy
- *dirnum* – počet podadresářů v adresáři

- *filenum* – počet souborů v adresáři

Pří průchodu tímto blokem dojde k nastavení:

```
memset(ADR(ManageFile.Data.FileListB[0]),0,SIZEOF(ManageFile.Data.FileLis);
memset(ADR(ManageFile.Data.DirListB[0]),0,SIZEOF(ManageFile.Data.DirListB);
```

tímto si vyčistíme pole od předchozí adresářové a souborové struktury.

Přechod do stavu `ManageFile.States := Read_Dir.`

Při chybovém výstupu přechod do stavu *ERROR.*

5.2.2 Read_Dir

Obsahuje funkční blok *DirRead*, který se stará o čtení podadresářů v adresáři a uložení do pole.

Vstupní parametry:

- *Enable* – povolení funkčního bloku
- *pDevice* – ukazatel na název zařízení, které bylo připojeno pomocí FB *DevLink*
- *pPath* – ukazatel na cestu k adresáři
- *entry* – nastaven na proměnou *Counter*
- *option* – nastavena hodnota fiDIRECTORY
- *pData* – proměnná *ReadData*
- *data_len* – velikost *ReadData*

Výstupní parametry:

- *status* – chybové kódy

Nastavením parametru *option* na *fiDIRECTORY* zajistíme čtení pouze adresářů.

V případě že výstupní *status = 0* kopírujeme jednotlivé nalezené adresáře do připraveného pole. Tento postup se opakuje, dokud není splněna podmínka *Counter < dirnum.*

Přechod do stavu `ManageFile.States := Read_File;`

Při chybovém výstupu přechod do stavu *ERROR.*

5.2.3 Read_File

Obsahuje funkční blok *DirRead*, který se stará o s přečtení souborů v adresáři a uložení do pole.

Vstupní parametry:

- *Enable* – povolení funkčního bloku
- *pDevice* – ukazatel na název zařízení, které bylo připojeno pomocí FB *DevLink*

- *pPath* – ukazatel na cestu k souboru
- *entry* – nastaven na proměnou *Counter*
- *option* – nastavena hodnota fiFILE
- *pData* – proměnná *ReadData*
- *data_len* – velikost *ReadData*

Výstupní parametry:

- *status* – chybové kódy

Nastavením parametru *option* na *fiFILE* zajistíme čtení pouze souborů.

V případě že výstupní *status = 0* kopírujeme jednotlivé nalezené soubory do připraveného pole. Tento postup se opakuje, dokud není splněna podmínka *Counter* < *filenum.*

V tomto stavu kontrolujeme a nastavujeme několik proměnných, které nám přichází z inicializační části (`InitCounter`), z kopírování souboru (`CopyFileChangeSection`) a mazaní adresáře (`MoveFileTmp`).

Přechod do stavu `ManageFile.States := Action;`

Při chybovém výstupu přechod do stavu *ERROR.*

5.3 Action

Tento stav je řídící jednotka celého programu. Probíhá odchytávání změny proměnných, pomocí kterých je zajištěn přesun do jednotlivých stavů.

Proměnné pro přesuny:

- aCreateFile – stav *Create_File* – vytváření souborů
- aDeleteFile – stav *Delete_file* – mazání souborů
- aCopyFile – stav *Copy_File* – kopírování souborů
- aMoveFile – stav *Move_File* – přesun souborů
- aRenameFile – stav *Rename_File* – přejmenování souborů
- aOpenDir – stav *Open_Dir* – vstup do adresáře
- aCreateDir – stav *Create_Dir* – vytvoření adresáře
- aDeleteDir – stav *Delete_Dir* – smazání adresáře
- aRenameDir – stav *Rename_Dir* – přejmenování adresáře
- aGoBackDev – stav *Return_Path* – přesuny zpět v adresářové struktuře
- aDevice_C – stav *SetupDevice* – nastavení zařízení na disk C
- aDevice_D – stav *SetupDevice* – nastavení zařízení na disk D
- aDevice_F – stav *SetupDevice* – nastavení zařízení na USB disk

Pomocí tohoto stavu kontrolujeme změnu stavu proměnné *Name,* ke které dochází při výberu adresáře či složky. Za podmínky *tmpName* <> *Name* dochází k přechodu do stavu `ManageFile.States := Dev_Unlink;`

5.4 CheckParameter

Stav při změně zachycené ve stavu *Action* podmínkou *tmpName <> Name*.

Pomocná proměnná *Exist* určuje další postup v tomto stavu. Pří vstupu má hodnotu *FALSE*. Postupně projdeme pole s adresáři a soubory. Při nalezení souboru či adresáře nastavíme *Exist = TRUE* a vrátíme se dostavu *Action* v opačném případě bude výstupem chybový stav.

5.5 Return_Path

Stav, který je volán při výběru aGoBackDev ve stavu *Action*.

Pomocí pole s uchovanými adresami průchodu `PathListDevA(B)` a proměnnou `PathCounterA(B),` která nám slouží k pohybu v poli, nastavíme parametr cesty předchozího adresáře a předáme ho funkčnímu bloku *DevLink*.

Vymažeme z pole aktuální adresu, na které se nacházíme.

Přechod do stavu `ManageFile.States := Dev_Unlink;`

5.6 Dev_Unlink

Obsahuje funkční blok *DevUnlink*, který se stará o odebrání připojeného disku.

Vstupní parametry:

- *Enable* – povolení funkčního bloku
- *Handle* – výstupní parametr momentálně připojeného disku v *DevLink*.

Výstupní parametry:

- *status* – chybové kódy

Přechod do stavu `ManageFile.States := Dev_Link;`

Při chybovém výstupu přechod do stavu *ERROR*.

5.7 Dev_Link

Obsahuje funkční blok *DevLink*, který se stará o připojení (vytvoření) disku.

Vstupní parametry:

- *Enable* – povolení funkčního bloku
- *pDevice* – ukazatel na název zařízení
- *pParam* – ukazatel na cestu k disku

Výstupní parametry:

- *status* – chybové kódy
- *handle* – výstupní parametr pro FB *DevUnlink*

Přechod do stavu `ManageFile.States := Dir_Info;`

Při chybovém výstupu přechod do stavu *ERROR.*

Přechod do stavu `ManageFile.States := Dir_Info;`

5.8 Správa disků

Stav, který je volán při výběru *aDevice_C, aDevice_D, aDevice_D* ve stavu *Action*.

Vymazání pole pro uchovávání průchodů adresáři na disku:

```
memset(ADR(ManageFile.Data.PathListDevA[0]),0,SIZEOF(ManageFile.Data.PathListDevA))
```

Dle výběru disku nastavit první index v poli na jeho kořenový adresář.

Přechod do stavu `ManageFile.States := Dev_Unlink;`

5.9 ERROR

Vstup do stavu ERROR je ze všech funkčních bloku, kde se výstupní parametr *status* nerovná kódům 0, 65535, 65534,

Jsou zde zachytávány kódy jednotlivých chybových stavů a zprávy jsou podávány formou proměnné *Error*, kde se vypíše kód chybového hlášení.

Případně je podávána zpráva pomocí proděné *Message* kde jsou informace podány formou textové zprávy.

V tomto stavu se také čeká na interakci od uživatele, který je povinen potvrdit informaci o chybě pomocí proměnné `ErrACK = TRUE.`

Funkčním blokům je nastaven parametr *enable* na FALSE.

Přechod do stavu `ManageFile.States := Action;`

6. Správa adresářů

Správa adresářů obsahuje funkční bloky, které vykonávají procesy potřebné k vytváření nových adresářů, přejmenování adresářů a mazání adresářů.

6.1 Open_Dir

Stav, který je volán při výběru *aOpenDir* ve stavu *Action*.

Pro vstup do adresáře je třeba nastavit nový parametr pro funkční blok *DevLink* a to pomocí spojení názvu vybraného adresáře a parametru s aktuálním umístěním. K tomu využijeme funkci *CONCAT*, která nám spojí dva textové řetězce.

```
ManageFile.Data.Parameter_Dev_A:=CONCAT(ManageFile.Data.PathListDevA[PathCo
unterA-1],CONCAT('\',ManageFile.Data.Name));
```

Nově vytvořený parametr je předán k nastavení do funkčního bloku *DevLink*.

Přechod do stavu `ManageFile.States := Dev_Unlink;`

6.2 Create_Dir

Stav, který je volán při výběru *aCreateDir* ve stavu *Action*.

Obsahuje funkční blok *DirCreate,* který se stará o vytvoření nového adresáře.

Vstupní parametry:

- *Enable* – povolení funkčního bloku
- *pDevice* – ukazatel na název zařízení, které bylo připojeno pomocí FB *DevLink.*
- *pName* – ukazatel na název nového adresáře

Výstupní parametry:

- *status* – chybové kódy

Při existenci adresáře se stejným názvem je výstupem chybový stav.

Přechod do stavu `ManageFile.States := Dir_Info;`

Při chybovém výstupu přechod do stavu *ERROR.*

6.3 Rename_Dir

Stav, který je volán při výběru *aRenameDir* ze stavu *Action.*

Obsahuje funkční blok *DirRename,* který se stará o přejmenování adresáře.

Vstupní parametry:

- *Enable* – povolení funkčního bloku
- *pDevice* – ukazatel na název zařízení, které bylo připojeno pomocí FB *DevLink.*
- *pName* – ukazatel na název adresáře
- *pNewName* – ukazatel na název nového adresáře

Výstupní parametry:

- *status* – chybové kódy

Při existenci adresáře se stejným názvem je výstupem chybový stav.

Přechod do stavu `ManageFile.States := Dir_Info;`

Při chybovém výstupu přechod do stavu *ERROR*

6.4 Delete_Dir

Stav, který je volán při výběru *aDeleteDir* ze stavu *Action.*

Obsahuje funkční blok *DirDelete,* který se stará o smazání prázdného adresáře.

Vstupní parametry:

- *Enable* – povolení funkčního bloku
- *pDevice* – ukazatel na název zařízení, které bylo připojeno pomocí FB *DevLink.*
- *pName* – ukazatel na název adresáře

Výstupní parametry:

- *status* – chybové kódy

Přechod do stavu `ManageFile.States := Dir_Info;`

V případě, že adresář není prázdný, dostaneme chybový stav *status =20799.* Přejdeme do stavu *WaitDeleteDir,* kde se čeká na rozhodnutí od uživatele, zda chce smazat adresář i s podadresáři.

Výběr je proveden pomocí proměnné *FolderDeleteConfirm.*

V případě potvrzení smazání adresáře `FolderDeleteConfirm = 1` přejdeme do stavu *Delete_Dir_All* v opačném případě do stavu *Action* a adresář bude zachován.

Při chybovém výstupu přechod do stavu *ERROR.*

6.5 Delete_Dir_All

Při uživatelském výběru `FolderDeleteConfirm = 1` ze stavu *Delete_Dir.*

Obsahuje funkční blok *DirDeleteEx,* pomocí kterého je zajištěno smazání adresáře i s podadresáři a soubory.

Vstupní parametry:

- *Enable* – povolení funkčního bloku
- *pDevice* – ukazatel na název zařízení, které bylo připojeno pomocí FB *DevLink.*
- *pName* – ukazatel na název adresáře

Výstupní parametry:

- *status* – chybové kódy

Přechod do stavu `ManageFile.States := Dir_Info;`

Při chybovém výstupu přechod do stavu *ERROR.*

7. Správa souborů

Správa souborů obsahuje funkční bloky, které vykonávají procesy potřebné k vytváření nových souborů, přejmenování, kopírování, přesouvání a mazání souborů.

7.1 Create_File

Stav, který je volán při výběru *aCreateFile* ze stavu *Action.*

Obsahuje funkční bloky *FileCreate,* který se stará o vytváření nových souborů.

Vstupní parametry:

- *enable* - povolení funkčního bloku
- *pDevice* - ukazatel na název zařízení, které bylo připojeno pomocí FB *DevLink*
- *pFile* - název souboru, který chceme vytvořit

Výstupní parametry:

- *status* – chybové kódy
- *ident* – identifikace vytvořeného souboru

Při existenci souboru se stejným názvem je výstupem chybový stav.

Přechod do stavu `ManageFile.States := cClose_File;`

Při chybovém výstupu přechod do stavu *ERROR.*

7.2 cClose_File

Stav, ve kterém se provádí zavření nově vytvořeného souboru pomocí funkčního bloku *FileClose*. Tento stav je nutnou podmínkou při volání FB *FileCreate.*

Vstupní parametry:

- *enable* - povolení funkčního bloku
- *ident* - identifikace otevřeného souboru

Výstupní parametry:

- *status* – chybové kódy

Přechod do stavu `ManageFile.States := Dir_Info;`

Při chybovém výstupu přechod do stavu *ERROR.*

7.3 Copy_File

Stav, který je volán při výběru *aCopyFile, aMoveFile* ze stavu *Action.*

Obsahuje funkční bloky *FileCopy,* který se stará o kopírování souboru.

Vstupní parametry:

- *enable* - povolení funkčního bloku
- *pSrcDev* - ukazatel na název zařízení, které bylo připojeno pomocí FB *DevLink*
- *pSrc* – ukazatel na název souboru
- *pDextDev* – ukazatel na název cílového zařízení na které budou soubor kopírován
- *pDest* – ukazatel na název souboru v cílovém zařízení
- *option* – nastavení možnosti pro přepisování již existujícího souboru

Výstupní parametry:

- *status* – chybové kódy

Při první průchodu probíhá kontrola názvu souboru, zda již tento název neexistuje v cílovém zařízení.

V případě že název existuje přecházíme do stavu *WaitCopyFile,* kde se čeká na potvrzení uživatele, zda chce soubor přepsat nebo vytvořit soubor s přídomkem –kopie.

Proces přesouvání souboru je volán pomocí *aMoveFile* ve stavu *Action,* kde se předá pomocná proměnná *MoveFileTmp.* Při přesouvání souboru je v případě úspěšného překopírování souboru volán stav *Delete_File* , který smaže soubor ze zdrojového zařízení.

Přechod do stavu `ManageFile.States := Dir_Info;`

7.4 Rename_File

Stav, který je volán při výběru *aRenameFile* ze stavu *Action.*

Obsahuje funkční bloky *FileRename,* který se stará o přejmenování souborů.

Vstupní parametry:

- *enable* - povolení funkčního bloku
- *pDevice* - ukazatel na název zařízení, které bylo připojeno pomocí FB *DevLink*
- *pName* – ukazatel na název souboru
- *pNewName* – ukazatel na název nového souboru

Výstupní parametry:

- *status* – chybové kódy

Při existenci souboru se stejným názvem je výstupem chybový stav.

Přechod do stavu `ManageFile.States := Dir_Info;`

Při chybovém výstupu přechod do stavu *ERROR.*

7.5 Delete_File

Stav, který je volán při výběru *aDeleteFile* ze stavu *Action.*

Obsahuje funkční bloky *FileDelete,* který se stará o smazání souborů.

Vstupní parametry:

- *enable* - povolení funkčního bloku
- *pDevice* – ukazatel na název zařízení, které bylo připojeno pomocí FB *DevLink*
- *pName* – ukazatel na název souboru

Výstupní parametry:

- *status* – chybové kódy

Přechod do stavu `ManageFile.States := Dir_Info;`

Při chybovém výstupu přechod do stavu *ERROR.*

8. Závěr

V bakalářské práci byl vytvořen program zajišťující:

1. Práci se soubory
2. Práci s adresáři
3. Ovládání úložných zařízení
4. Uživatelská interakce

Ad 1: Použitím knihovních funkcí bylo zjištěno vytváření souborů dle zadaného názvu, kopírování a přesouvání souborů, přejmenování souborů a mazání souborů. Vše podléhá kontrole, zda soubor existuje a zabraňuje vytváření či přepisování již existujících souborů.

Ad 2: Použitím knihovních funkcí bylo zajištěno vytváření, přejmenování a mazání adresářů. Všechny procesy podléhají kontrole existence a je zajištěno, aby nedošlo k vytváření již existujících adresářů a nedošlo ke smazání adresáře obsahujícího jinou strukturu.

Ad 3: Pomocí jednoduchého rozhraní můžeme ovládat přístup k zařízením, se kterými můžeme pracovat.

Ad 4: Pro dokončení některých procesů jsou vyžadována uživatelská rozhodnutí. Při kopírování a přesouvání souborů jde o potvrzení, zda uživatel chce přepsat již existující soubor nebo ho pouze přejmenovat. V případě že se uživatel snaží smazat adresář, který obsahuje jiné adresáře či soubory, je zde na uživateli aby potvrdil, zda opravdu chce celý adresář smazat.

Práci by bylo do budoucna dobré rozšířit o grafické uživatelské prostředí.

Jistě by prospělo optimalizovat práci s využitím dynamicky alokované paměti, pro uchovávání informací o souborech a adresářích. Momentálně je to řešeno formou statického pole, které je pro menší počet prvků dostačující.

Seznam literatury a dalších pramenů

[1] AUTOMATION, B&R. Controls - training text. Austria : [s.n.], 2008. 205 s.

[2] BERNECKER + RAINER INDUSTRIE-ELEKTRONIK GES.M. B. H. B&R Help: FileIO. 2013. vyd. 2013.

[3] ŠMEJKAL, Ladislav a Marie MARTINÁSKOVÁ. PLC a automatizace. 1. díl. Praha: BEN, 1999. ISBN 80-8605--58-9.

[4] JOHN, Kharl-Heinz; TIEGELKAMP, Michael. IEC 61131-3 Programming Industrial Automation Systems : Concepts and Programming Languages, Requirements for Programming Systems, Decision - Making Aids. 2nd Edition. NewYork : Springer, 2010. 390 s. ISBN 978-3-642-12015-2.

Příloha č.1

- Bakalarska_prace_2015_Lukas_Kin.pdf
- Bakalarska_prace_2015_Lukas_Kin_program
    - ManageFile.zip